Preface

Peer-to-peer (P2P) technology, or peer computing, is an emerging paradigm that is now viewed as a potential technology that could re-architect distributed architectures (e.g., the Internet) and processing. In a P2P network, all participating computers (or nodes) have equivalent capabilities and responsibilities. The nodes can directly exchange resources and services between themselves without the need for centralized servers. The nodes can collaborate to perform tasks by aggregating the pool of resources (e.g., storage, CPU cycles) available in the P2P network. The distributed nature of such a design provides exciting opportunities for new killer applications to be developed.

P2P computing distinguishes itself from traditional distributed computing in several important aspects. First, P2P applications reach out to harness the outer edges of the Internet and consequently involve scales that were previously unimaginable. Second, P2P by definition, excludes any form of centralized structure, requiring control to be completely decentralized. Finally, and most importantly, the environments in which P2P applications are deployed exhibit extreme dynamism in structure, content and load. The topology of the system typically changes rapidly due to nodes voluntarily joining and leaving the network or due to involuntary events such as crashes and partitions. The load in the system may also shift rapidly from one region to another, for example, as certain files become "hot" in a file sharing system; or the computing needs of a node suddenly increase in a grid computing system.

The scale and dynamism that characterize P2P systems require traditional distributed technologies to be reexamined. A paradigm shift that includes self-reorganization, adaptation and resilience is also called for. In recent years, there has been a proliferation of research efforts to design P2P systems and applications. This book attempts to present the *technical* challenges offered by P2P systems, and the efforts that have been proposed to address them. The purpose of this book is to provide a thorough and comprehensive review of recent advances on routing and discovery methods, programming models, security, accountability, anonymity and P2P systems and projects. Besides surveying existing methods and systems, the book also compares and evaluates some of the more promising schemes.

The need for such a book is evident. It provides a single source for practitioners, researchers and newcomers on the state-of-the-art in the field. For practitioners, this book explains best practice, guiding selection of appropriate techniques for each application. For researchers, this book provides a foundation for development of new and more effective methods. For newcomers, this book is an overview of the wide range of advanced techniques for realizing effective P2P systems. This book can also be used as a text for an advanced course on Peer-to-Peer Computing and Technologies, or as a companion text for a variety of courses including courses on distributed systems and grid and cluster computing.

Organization of the Book

This book consists of ten chapters. Besides the first chapter that sets up the context and the last chapter that concludes with directions on the future of P2P, each of the other eight chapters is essentially self-contained and focuses on one aspect of P2P computing. These eight chapters can thus be read and used on their own independently of the others.

In Chapter 1, we provide background on P2P computing in general. We discuss the characteristics of P2P systems that distinguish them from distributed systems. This chapter also looks at the benefits and promises of P2P, and some of the applications that will benefit from P2P computing. It also examines the issues in designing P2P systems and sets the stage for subsequent chapters.

Chapter 2 presents the various architectures of P2P systems. On one extreme, we have P2P systems that are supported by centralized servers. On the other extreme, pure P2P systems are completely decentralized. Between these two extremes are hybrid systems where nodes are organized into two layers: the upper tier "super" nodes act as servers for lower tier nodes. We also compare these different architectures. We also look at how peers are defined - statically or dynamically. Support for dynamic reorganization of peers allows communities to be formed based on some common interests among nodes. We will also examine how nodes that are more powerful can be exploited to shoulder more responsibilities. Issues on incentives and fairness will also be addressed.

In Chapter 3, we focus on the issue of searching. There are several modes in which searching can be performed. First, a query node can broadcast queries to all nodes. Second, the query can be directed to nodes that are more likely to contain useful information first. This requires nodes to organize thier peers based on some optimization criterion. Third, hashing techniques can be applied. We will also look at how load-balancing can be realized in the hash-based category. Each of these techniques call for different metadata to be maintained.

Chapter 4 presents techniques to perform complex queries. Besides simple keyword search, there is an increasing need to support more semantic-based queries for database and multimedia applications. These include partial match queries, range and join queries, and queries involving high-dimension vectors. We also looked at how distributed queries are optimized and processed in P2P context.

Replication and caching are very effective mechanisms that can bring the data/results closer to the users to improve performance. However, in P2P environment, it becomes much harder to control the optimal degree of replication as well as consistency. Chapter 5 presents the issues that need to be addressed, and examines some of the existing solutions. In particular, we will look at techniques that manage replicas/cache dynamically.

In Chapter 6, we look at programming models that are suitable for P2P environments. Most of the existing P2P systems lack an adequate parallel programming model. Moreover, unlike parallel programming systems, the unique features of P2P environment such as dynamic resource discovery and fault-tolerance and availability should be considered to develop an integrated environment optimized for parallel computing. We shall examine the design of some of the existing parallel programming models, including P^3 and Teaq.

Before P2P can be widely accepted by users, there are several other issues that need to be addressed: trust, privacy, anonymity, accountability, reliability and security. These issues are discussed in Chapters 7 and 8. In Chapter 7, we focus on accountability, trust and reputation. Here, we look at techniques that automate the collection and processing of information from previous queries to help users assess whether they can trust a server with a new query. We also discuss methods to prevent users from taking advantage of the system by freeloading off the resources contributed by a few. Techniques that authenticate third-party data publication will also be examined in this chapter.

Chapter 8 focuses on security, privacy and anonymity issues. We look at techniques that are designed to support anonymity to protect both the users that disseminate the data, as well as nodes that store the data. Techniques to protect the privacy of information and users are also discussed. Finally, we discuss techniques that have been designed to secure data as well as the P2P environment from attacks.

Chapter 9 presents some representative P2P systems and applications that have been deployed. We will look at how different applications and requirements drive the design and architecture of the systems. For example, missing some data may be acceptable in a music sharing application but not so in a database application (where data integrity and consistency are important). We will present systems that support data distribution, search, code distribution and collaboration.

Finally, in Chapter 10, we suggest promising research topics that deserve further attention. In particular, integrating agent technology and P2P tech-

nology has the potential to overcome the problem of weaker nodes. Another important direction is the exploitation of P2P technology in mobile and wireless computing environment. Yet another extension is to integrate XML into P2P context to facilitate more effective information exchange.

Panagiotis Kalnis,
Beng Chin Ooi,
Kian-Lee Tan,
Aoying Zhou

1. Introduction

Peer-to-peer (P2P) computing has re-made itself to become a promising paradigm for distributed computing. This twenty-year-old technology was deployed in USENET in 1979 and FiDoNet in 1984. At that time, the number of computer users is relatively small and P2P applications are less user-friendly. Moreover, users fail to recognize the benefits of the technology. However, several trends have re-focused the attention of researchers on this technology. First, the Internet has allowed a large number of computers to be connected. Second, the Internet has also provided an avenue for users to share and disseminate their data in a user-friendly manner. Third, "killer" P2P applications have surfaced. For example, the Napster [20] MP3 music file sharing applications served over 20 million users by mid-2000. As another example, the SETI@home [21] program has accumulated over 500,000 years of CPU time through more than 2 million users.

In this chapter, we provide background on P2P computing in general. We discuss the characteristics of P2P systems that distinguish them from traditional distributed systems. This chapter also looks at the benefits and promises of P2P, and some of the applications that will benefit from P2P computing. It also examines the issues in designing P2P systems.

1.1 Peer-to-Peer Computing

Peer-to-peer (P2P) computing is essentially a model of how we (people) interact in real life. We deal directly with one another when we want to. Very often, when we need something, we ask our peers who may in turn refer us to their peers. P2P technologies enable us, through our computers, to carry our interactions into cyberspace and to continue to deal with one another as we do in the real world.

In a P2P network, all participating computers (or nodes) have equivalent capabilities and responsibilities. The nodes can directly exchange resources and services between themselves without the need for centralized servers. The system can aggregate resources and data from nodes to accomplish a task. Figure 1.1 illustrates how P2P computing operates for data sharing applications. Each node typically maintains some metadata that facilitates searching. Moreover, each node will indicate to the application the types of

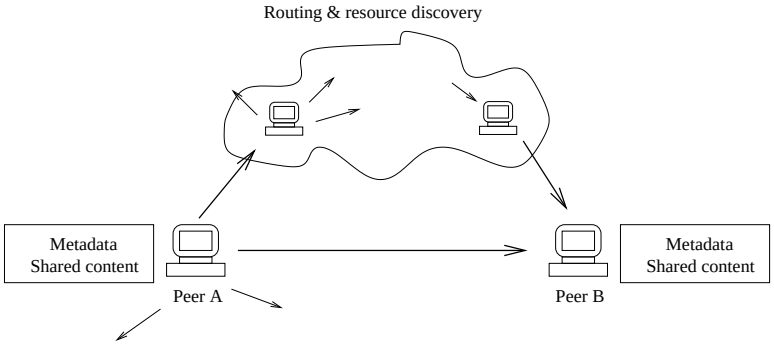

Fig. 1.1. Peer-to-peer computing. Peer A requests for some data that Peer B has. However, Peer A has to first locate Peer B through other peers in the P2P network. Once Peer B is located, Peer A deals directly with Peer B.

data that it will contribute share with the community. A query for data will involve a resource discovery process that routes the query around the network to nodes that own the data. There may be more than one node that contains the data. The query node can then directly communicate with the owners to acquire the data.

P2P computing is really distributed computing. Both share the same set of issues that distributed computing researchers have been addressing over the years (e.g., security, trust, anonymity, fault tolerance, scalability, distributed query processing and coorination). However, P2P computing distinguishes itself from traditional distributed computing in several aspects. Some of the more important ones are:

1. Symmetric role. Each participating node in a P2P system acts as both a server and a client. In fact, each node installs a single package that encompasses both client and server code. As such, a node can issue query (like a client) and serves requests (like a server).
2. Scalability. P2P applications reach out to harness the outer edges of the Internet and consequently involve scales that were previously unimaginable. The protocols do not require "all-to-all" communication or coordination.
3. Heterogenity. The P2P system is highly heterogeneous in terms of the hardware capacity of the nodes - a node may be a very slow machine and another may be a high-end super computer.
4. Distributed control. P2P by definition, excludes any form of centralized structure, requiring control to be completely decentralized.
5. Dynamism. The environments in which P2P applications are deployed exhibit extreme dynamism in structure, content and load. The topology of the system typically changes rapidly due to nodes voluntarily joining and leaving the network or due to involuntary events such as crashes and partitions.

The scale and dynamism that characterize P2P systems requires traditional distributed technologies to be reexamined. A paradigm shift that includes self-reorganization, adaptation and resilience is also called for.

1.2 Potential, Benefits and Applications

P2P computing has tremendous potential to meet many organizational and personal needs. It not only leverages on computing resources (within an organization or in the internet) without excessive cost, it also allows information to be disseminated widely. Furthermore, it enables the owners of content to exercise full control over their data.

In recent years, there has been a proliferation of research efforts to design P2P systems and applications. These applications can be broadly divided into two categories: *resource sharing* and *data sharing*. In resource sharing, applications allow enterprises or individuals to leverage on available (idle or otherwise) CPU cycles, disk storage and bandwidth capacity within the P2P network. P2P computing enables harnessing of underused resources to perform tasks that would otherwise require a much more expensive machine such as a super computer. Similarly, data storage devices could be exploited to create a wide area storage network, and to push the data closer to the users. In data sharing, applications allow users to access, modify and exchange data in a flexible manner. Some successful applications include:

- *Scientific computation.* Many scientific research projects involve extensive computation that typically require massive supercomputers. However, with P2P techology, we can now exploit the large number of computers (e.g., PCs) participating in the P2P network to perform the task. This not only saves cost, but also makes more effective use of the large number of idling computers sitting around. The most notable project is the Search for Extraterrestrial Intelligence (SETI) at Home (SETI@home) project [21]. The goal of SETI is to detect aliens and intelligent life outside the Earth. To make use of less powerful computers, SETI splits each computational task into manageable *work units*. Each home PC operates on a work unit, and when it has completed its assignment, it picks up another work unit. In this way, SETI is able to develop the "world's most powerful computer". For example, as reported in [7], SETI@home is faster than ASCI White, at less than 1% of the cost. Moreover, in a typical day, SETI@home clients (i.e., the PCs) process about 700,000 work units, which works out to over 20 TFLOPS. The success of SETI will prompt more compute intentive projects (whose tasks can be split into sub-tasks with little or no interdepence and the ratio of communication overhead and computation is low) to exploit PCs within or without an organization, e.g., the Stanford University has set up the Folding@Home project that studies protein folding, misfolding, aggregation and related diseases [1].

- *Instant Messaging.* People communicate and converse to acquire and spread knowledge in real life. However, conversations can take place within the Internet through *conversational technologies.* This allows "meetings" to be organized among friends and associates in the Internet. Instant messaging (IM) is one such technology that enables users to locate their peers, provides a P2P communication path, and even offers an informal status of a peer's availability. Through IM platforms, users can compose messages and transmit files to one or more peers that are online. Typically, peers are connected to mediating servers who are responsible for negotiating the delivery and receipt of their clients' messages with other servers. The message is routed from node to node until the server closest to the recepient is reached, who will then deliver the message. Once connected to their servers, computers at the network's edge can establish real-time conversations with any other peers. Some widely used IM solutions are ICQ [14], AOL instant messaging, Yahoo instant messenger and Jabber [18, 3].
- *Digital content sharing.* The Internet is essentially an asymmetric shared content repository, where there are a small number of content providers (servers) but a large number of content consumers (end users). P2P technology overcomes this asymmetry by enabling users to act as a producer as well as a consumer. Essentially, a request for some digital content is passed from peer to peer, and as each peer is traversed it will pass back the requested content, if any, to the query node (or through the peer that forwarded the request); it will also forward the request to other peers. In this way, a peer contributes his/her content to the P2P network. Such content sharing not only allows owners to have control over their content, it also removes any single point of failure. Examples of P2P platforms that support content sharing are Gnutella [13, 16], Freenet [12, 17], Free Haven [9] and Publis [22].
- Distributed databases. Content sharing can be taken a step further by allowing local databases (stored in MySQL or Microsoft Access) to be shared. For example, in health care domain, hospital specialists typically have a group of patients that are solely under their care. While some patient data are stored in a centralized server of the hospital (e.g., name, address, etc), other data (e.g., X-rays, prescription, allergy to drugs, history, reaction to drugs, etc) are typically managed by the specialists on their PCs. For most of these patients, the specialists are willing to share their data, but there are always some cases that they are unwilling to share for different reasons (e.g., part of his research program on a new drug, etc). By making the sharable patient data available to other specialists, it allows them to look for other patients who may have similar symptoms as their own patients, and hence can help them in making better decisions on the treatment (e.g., drugs to prescribe, reactions to look out for, etc). As another example, in life sciences, the discovery of new proteins necessitates complex analysis in order to determine their functions and classifications. The main tech-

nique that scientists use in determining this information has two phases. The first phase involves searching known protein databases for proteins that "match" the unknown protein. The second phase involves analyzing the functions and classifications of the similar proteins in an attempt to infer commonalities with the new protein. While there are several known servers on genomic data (e.g., GenBank, SWISS-PROT and EMBL), there are many more data that are produced each day in the many laboratories all over the world. These scientists create their own local databases of their newly discovered proteins and results, and are willing to share their findings to the world!

- *Entertainment.* P2P technology established its roots in entertainment with the success of Napster [20]. The P2P system potentially provides a gigantic repository of music and video collection aggregated from individual user in the network. While these are essentially file sharing applications, P2P also fit well for interactive gaming over the internet. Each peer can store, manipulate and process complex models involving 3D graphics. The communication overhead between gaming peers can be minimal, e.g., a few message exchange may involve significant local computation and refreshing of the screen display (e.g., on how troops in a battle may be deployed). Examples of Internet P2P gaming platforms the Net-Z [5] and Star Craft [6].
- *Collaborative caching and storage.* Computers in a P2P system can contribute storage to enable content to be replicated and cached in different parts of the network. Such an environment offers many advantages. First, content can be brought closer to users that need them. For example, in e-learning applications, an education center can minimize remote accesses to course content (and hence minimize bandwidth consumption) by caching materials that are frequently needed on local nodes. Similary, internet accesses within an enterprise can exploit the local cache within each computer to share content that are common to most users [23]. Data warehouse is yet another practical application where caching is beneficial especially since the content of a warehouse is only updated periodically [15]. Second, a larger storage pool can facilitate anonymity as it becomes harder to identify the original source of content if multiple copies exist in the network. Third, availability and security can also be enhanced. For example, Publis employs secret sharing methods to distribute content *shares* to multiple nodes such that a subet of these shares can be used to reconstruct the content [22].
- *Collaborative work environments.* Today's work environments involve people who may be geogrphically dispersed. As such, it is critical for net-based collaboration tools to be developed to facilitate cooperation. P2P technology lends themselves well for cooperative collaboration environments. Here, a *collaboration* or *virtual space* will be created for the team members to interact and work together on project in real time. Shared content (e.g.,

documents and software) may be modified by any user, and automatically synchronized for consistency. Groove [2] and Magi [4] are examples of two P2P collaboration platforms.

1.3 P2P Enabling Technologies

P2P is not a panacea [19]. While it offers great potential and promises, there are many challenges that have to be addressed before its full potential could be realized. Some of these are:

- *Availability.* Because nodes can join and leave the P2P network, the system is essentially unpredictable. A resource (data or service) may be available at some time but not at others. These resources are available only when the nodes are connected to the network. As such, critical data or services may not be available when they are needed. Therefore, for a given query, the answer may be incomplete, and may also be different at different times. Mechanisms that replicate data or services can, to some extent, alleviate this problem.
- *Performance.* A same query will also experience different performance at different time depending on nodes that are connected and the network topology at time of query. Here, again, replication and caching may be useful as it brings the data closer to the query nodes. Mechanisms that load-balance the system will be very useful, for example, nodes that are more powerful may be exploited to perform a heavier load (recognizing that a node is powerful is a challenge!).
- *Integrity.* In a P2P environment, data may be replicated and cached in many nodes. It is hard to maintain the integrity and consistency of the data. There is a need to remove outdated copies or to refresh them. Techniques to validate or certify copies are also important especially since it is easy to proliferate content that misrepresent information. In particular, how to ascertain that a set of answers that are returned from a node is complete is very challenging.
- *Routing and resource discovery.* The main operation in a P2P environment is to be able to locate data or resources. At one extreme, we can employ a Gnutella-like mechanism [13] that broadcasts a query from a query node to its peers, who in turn will relay the message to their peers and so on. Such a method is simple, does not require any meta-data to be retained and can potentially reach to a large number of peers in the network. However, flooding the network with queries is bandwidth inefficient. Moreover, a large amount of resources are expended to evaluate the query - even peers that do not contain the results. At the other extreme, each peer can store some metadata that can direct the search for data/resource to the peers that contain the data/resource. The challenge, however, is to determine the type of metadata necessary for effective searching. Moreover, the need to

maintain the metadata can be complicated by peers' frequent connection and disconnection from the network. As such, there is a need to design effective and efficient data/resource discovery mechanisms.

- *Complex query processing.* Most of the existing P2P systems support simple queries such as keyword search. However, to support more applications such as databases, there is a need to design techniques for complex query processing. For example, there is a need to study how data stored in multiple relations relations can be combined efficiently. This calls for novel join algorithms to be developed. We also expect approximate query processing to play a more important role in P2P environments. In this case, there is a need to provide an indication on the quality of the approximate answers.
- *Replication and caching.* As noted above, replication and caching techniques are very important in P2P environments. However, techniques employed in distributed environments are not directly applicable because the topology of the network is continuously changing. It is much more difficult to control replicas and to refresh any cache data. Novel mechanisms are needed to optimize the allocation of replicas or cache data. Morever, as communities may shift from one region to another, techniques that can dynamically moves the cache content to new region may be useful.
- *Programming model.* Most of the existing P2P systems lack an adequate parallel programming model. Moreover, unlike parallel programming systems, the unique features of P2P environment such as dynamic resource discovery and fault-tolerance and availability should be considered to develop an integrated environment optimized for parallel computing. New programming models must be developed to fully exploit the potential of P2P computing.
- *Security.* P2P systems present interesting security problems. First, P2P applications could have security holes. By allowing other nodes to access a node's content/service through these applications, the node is vulnerable to attack. Second, a node (or even the entire network) could also be vulnerable to denial-of-service (DoS) attack as a malicious node can flood the node (or network) with queries. Such attacks are much harder to detect since these are at the application level. These call for novel methods to restrict access (access control mechanisms) to resources and data as well as techniques to detect application level DoS attacks.
- *Anonymity, Trust and Accountability.* There are other critical issues that may be specific to applications. For example, some applications may require support for anonymity. However, as noted in [8], providing anonymity may conflict with design goals for P2P systems. There is a need to design schemes that balance these goals. Another issue is the need for trust/reputation management. This is because the open and anonymity of P2P network leads to a complete lack of accountability. Effective reputation systems need to be developed to assure one of the quality of answers/service.

– *Incentives and Fairness.* For P2P system to be successful, there must be incentives for nodes to participate and contribute to the community. For example, a node may find itself being swamped by requests for some data that it has cached; without incentives, it may decide to leave the network. On the other hand, there may be nodes that are exploiting the system resource while contributing very little in return. Some mechanisms should be developed to ensure fairness in the system.

From the above discussions, it is clear that P2P computing offers tremendous amount of opportunity for research and development. This book is devoted to dealing with most of these issues and to review the various approaches that have been adopted in the literature.

1.4 P2P vs Grid Computing

Before leaving this chapter, we would like to compare P2P with Grid computing [11, 10]. Grid computing has emerged recently with the intent of scaling the system performance and availability by sharing resources. Like P2P computing, Grid Computing has been popularized by the need for resource sharing and consequently, it rides on existing underlying organizational structure. However, there are differences that distinguish the two.

First, the grid network involves higher-end resources as compared to edge level devices in the P2P network. While the former requires large amount of money to be pumped in, the latter can tap into existing resources that are idling and hence require less upfront cost commitment.

Second, the participants in the Grid network are organizations which agree in good faith to share resources with a good degree of trust, accountability and common understanding; membership can be rather exclusive and hence the number of participants is usually not large. The common platform for sharing is usually clusters that have been demonstrated to be cost effective to supercomputing, and together they provide an enormous amount of aggregated computing resources. In contrast, the participants of the P2P network are mainly end-users and the platform of sharing is mainly individual Personal Computer (PC). However, due to the mass appeal, the network grows in a much faster rate and may scale up to thousands of nodes. Because of the loose integration, it is more difficult and critical to manage trust, accountability and security.

Third, the Grid network is very much well structured and stable. As a result, resource discovery is less of an issue. On the contrary, P2P network is very unstable - nodes can join and leave the network anytime. This complicates the design of resource discovery mechanisms. Nodes that leave the network may mean some directories may be temporarily "unavailable".

Fourth, Grid computing can exploit traditional distributed query processing techniques and ensure that answers are complete. In contrast, nodes in

the P2P network containing data may not be connected at the time of query, answers are likely to be incomplete.

Finally, computational grids are largely set up in anticipation of resource intensive applications, e.g. BioGrid for bioinformatics. On the other hand, "killer" applications have surfaced in P2P naturally as can be seen from the success of Napster [20] and SETI@home [21].

In summary, we believe Grid computing will continue to play an important role in specialized applications, although architecturely, Grid computing can be considered a special case of P2P computing, where each participating node has a larger capacity and collaboration is more constrained and organized. Notwithstanding, we believe P2P technology is more "user friendly" in the sense that it allows users (particularly those at the edges) to share their resources and information easily and freely. P2P also offers more research challenges in view of the scale and unstability of the network.

1. Folding@home. In *http://folding.stanford.edu/*.
2. Groove. In *http://www.groove.net/*.
3. Jabber. In *http://jabber.org*.
4. Magi. In *http://www.endeavors.com/*.
5. Net-Z. In *http://www.proksim.com/*.
6. Star Craft. In *http://www.blizzard.com/*.
7. D. Anderson. SETI@home. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 5)*, pages 67–76. O'Reilly & Associates, 2001.
8. N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In *ICDE'2003*, 2003.
9. R. Dingledine, M. Freedman, and D. Molnar. Free Haven. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 12)*, pages 159–187. O'Reilly & Associates, 2001.
10. I. Foster and A. Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.
11. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.
12. Freenet Home Page. *http://freenet.sourceforge.com/*.
13. Gnutella Development Home Page. *http://gnutella.wego.com/*.
14. ICQ Home Page. *http://www.icq.com/*.
15. P. Kalnis, W.S. Ng, B.C. Ooi, D. Papadias, and K.L. Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *ACM SIGMOD 2002*, pages 25–36, 2002.
16. G. Kan. Gnutella. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 8)*, pages 94–122. O'Reilly & Associates, 2001.
17. A. Langley. Freenet. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 9)*, pages 123–132. O'Reilly & Associates, 2001.
18. J. Miller. Jabber. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 6)*, pages 77–88. O'Reilly & Associates, 2001.
19. D. Moore and J. Hebeler. *Peer-to-peer: Building Secure, Scalable, and Manageable Networks*. McGraw-Hill/Osborne, 2002.
20. Napster Home Page. *http://www.napster.com/*.
21. SETI@home Home Page. *http://setiathome.ssl.berkely.edu/*.
22. M. Waldman, L. Cranor, and A. Rubin. Publis. In *Peer-to-Peer : Harnessing the Power of Disruptive Technologies (Chapter 11)*, pages 145–158. O'Reilly & Associates, 2001.
23. X. Wang, W.S. Ng, B.C. Ooi, K.L. Tan, and A. Zhou. BuddyWeb: A p2p-based collaborative web caching system. In *Proceedings of the International Workshop on Peer-to-Peer Computing (LNCS 2376)*, pages 247–251, Pisa, Italy, 2002.